University of Burgundy

Image Processing

(Lab Report-2) :: Professor: Dr.SIDIBE

Student: VAMSHI KODIPAKA

2.1 Function Implementation of Image :: Translation

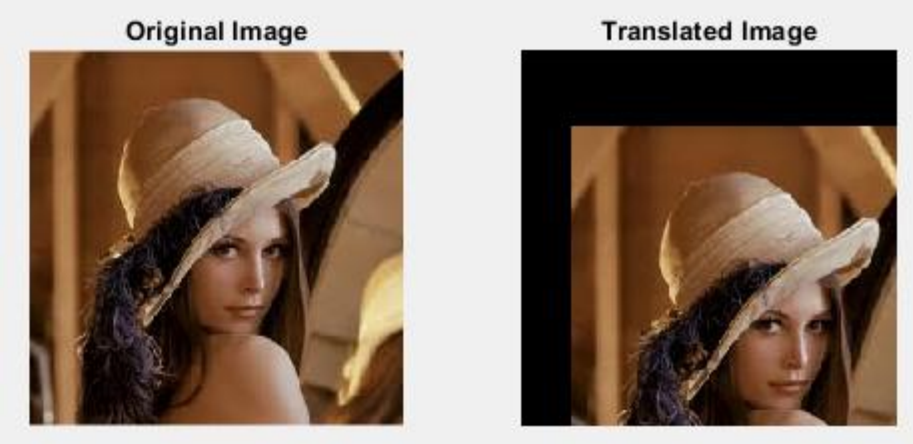
Explanation: Shift the image to ‘ty’ pixels horizontally and Shift the image to ‘‘tx’ pixels vertically

Implementing the Translation: myTranslation.m

Equation for Translation:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

Reading input images: Lena.jpeg **Output:**



Function working:

1. **Read input image lena.jpeg. Enter inputs tx=45 and ty=30**
2. **Shift tx units by rows ant ty units by columns**
3. **Store it as the output. Display now.**

NOTE: We will lose tx and ty units on opposite sides of image by translation, if we want to display the image in same pixels of that of an input image size.

2.2 Function Implementation of Image :: Rotation

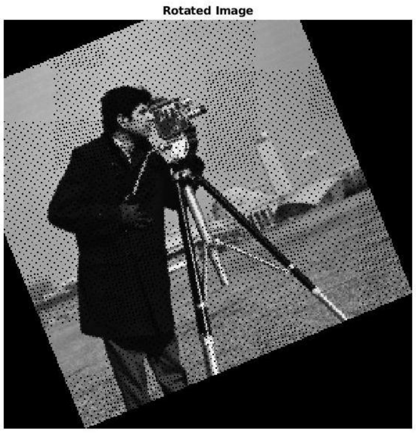
Explanation: Take the center of the image and rotate the image to ‘theta’ angle.

Implementing the Translation: myRotation.m

Equation for Rotation:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix} + \begin{pmatrix} x_c \\ y_c \end{pmatrix}$$

Reading input images: cameraman.tif   Output:



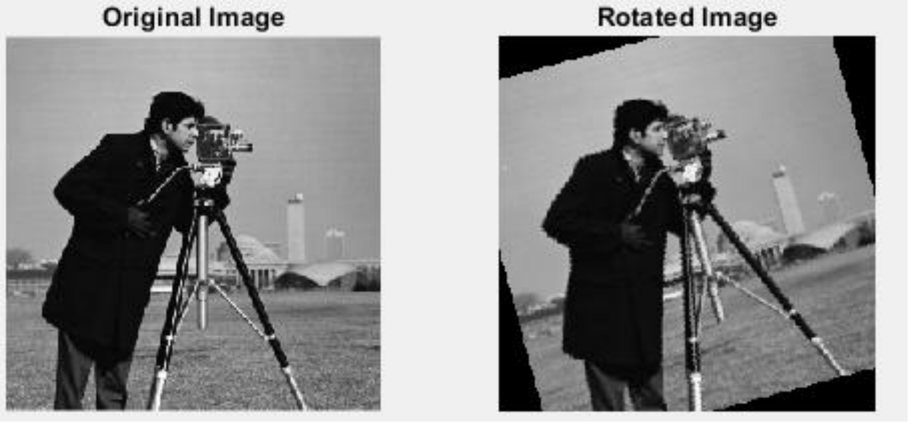
Function working:

1. Read input image cameraman.tif.Enter inputs theta= -30 degrees.
2. Take the center of the image . Rotate it.
3. Store it as the output. Display now.

2.2.1 Interpolation

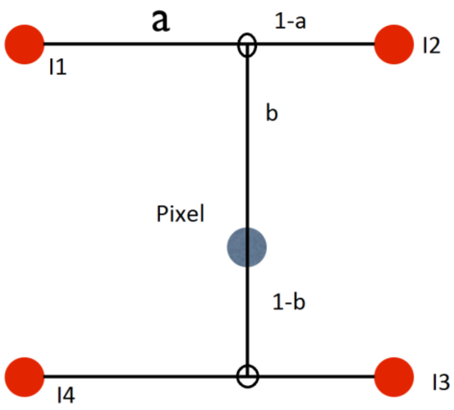
Interpolation: Rotation doesnot lead to perfect mapping of pixels instead leaves gaps in between the pixels after rotation. So we accumulate the surrounding pixels inorder to fill the gaps.

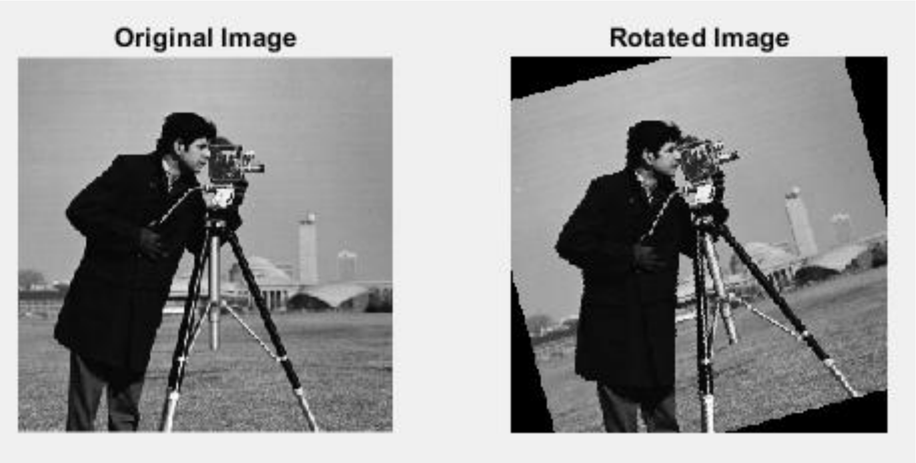
Nearest Neighbor: Take the average of all the corner pixel intensities to interpolate.(top, bottom, left,right)



Bilinear Transformation:

1. Set the reference pixel to rotate. Suppose take a pixel intensities I1,I2,I3,I4 respectively around four corners of the reference pixel.
2. Calculate the reference pixel’s distances along x and y reference axes(Let, say a,b).
3. Then we find linear relation between Intensities and pixels in one row of successive pixels.
4. We do this step twice for top and bottom corner pixel intensities.(w.r.t. opposite pixel intensities)
5. Then we take linear combination for resulting hollow dotted pixels.





NOTE: We will definitely find bilinear transformation accumulative than the nearest neighbor technique.

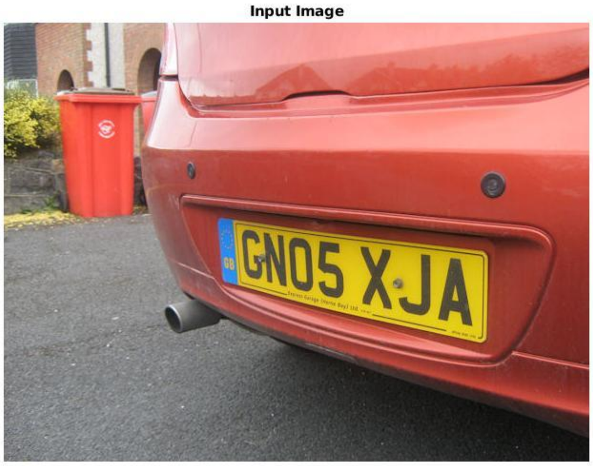
3. Function Implementation of Image :: Projective Transformation

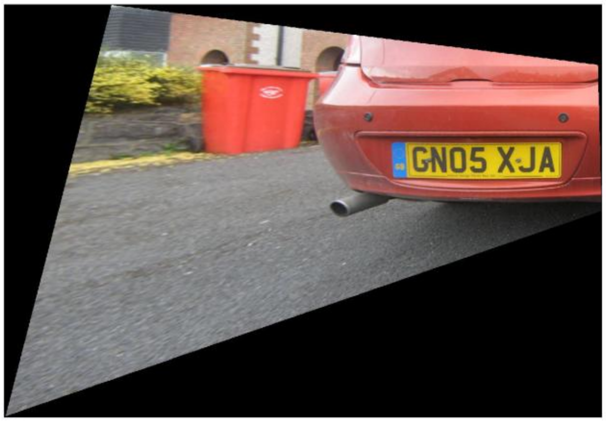
Explanation: Camera captures 3D object and uses perspective mapping on the object to corresponding points of the image and orientation. We align images more flat that obtained from various view points.

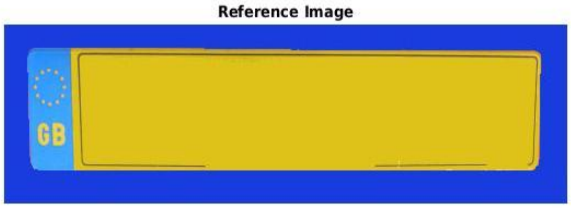
Implementing the Translation: projectiveTransform.m

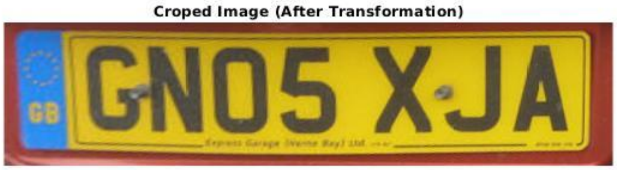
Function working:

1. Read input image plate_side.jpg.
2. Using pointer select the pixels points for the extraction of the input.
3. Select the pixels points for the reference image plate_reference.jpg.
4. We then display the projective transformation of the input.
5. Crop it and display the output image.









4. Function Implementation of Image :: Procustes Analysis

Explanation: We perform fitting of one image over the dimension of the other image.

Implementing the Translation: myProcrustes.m

Function working:

1. **Load star pattern.mat file**
2. **Find the procrustes matrix of the given input.**
3. **Take centroid of input to base pixels by Translation.**
4. **Scale the input pixels to the base image size.**
5. **Then rotate it to align the set of points in both the images.**
6. **Repeat it same for handpoints also.**

Equation for Translation:

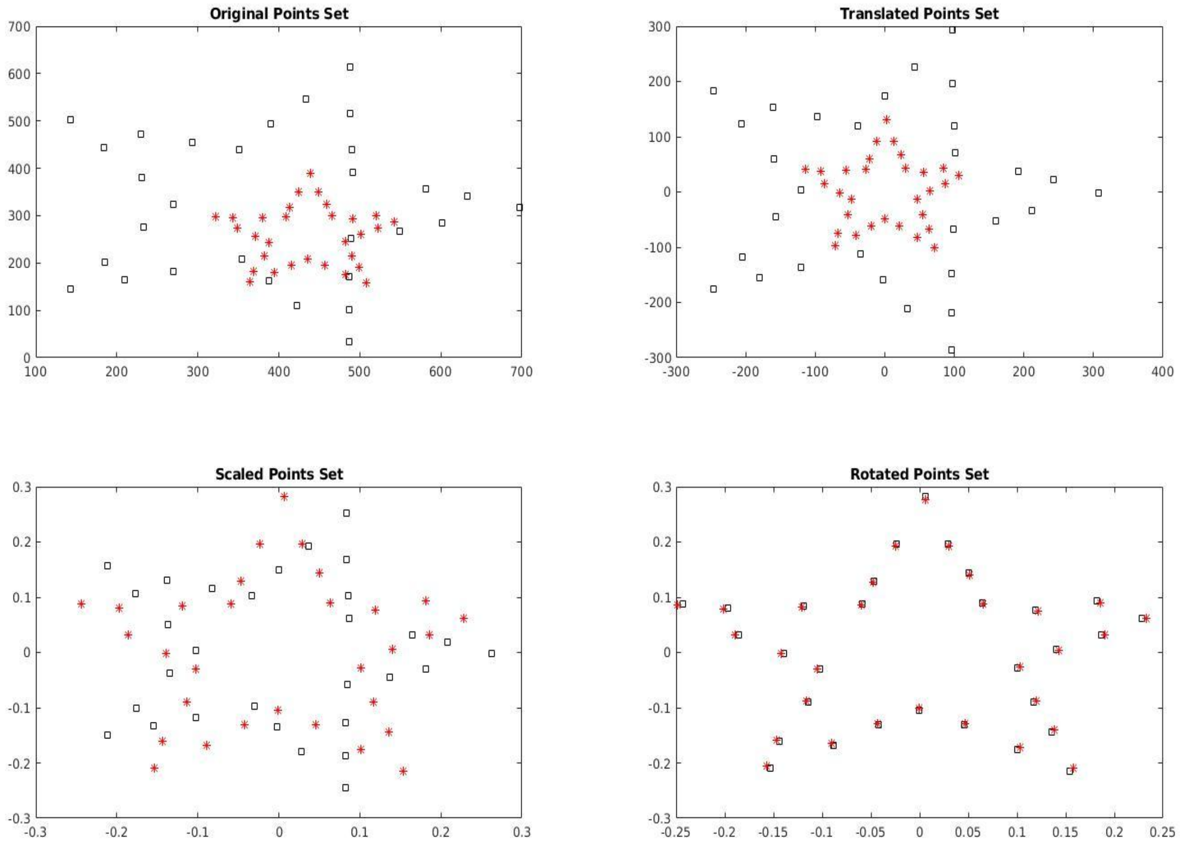
$$\mathbf{x}_i \rightarrow \mathbf{x}_i - \bar{\mathbf{x}},$$

Equation for Scaling: **we calculate the equation for ::: norm of x = {x_i/sqrt(x_i)}**

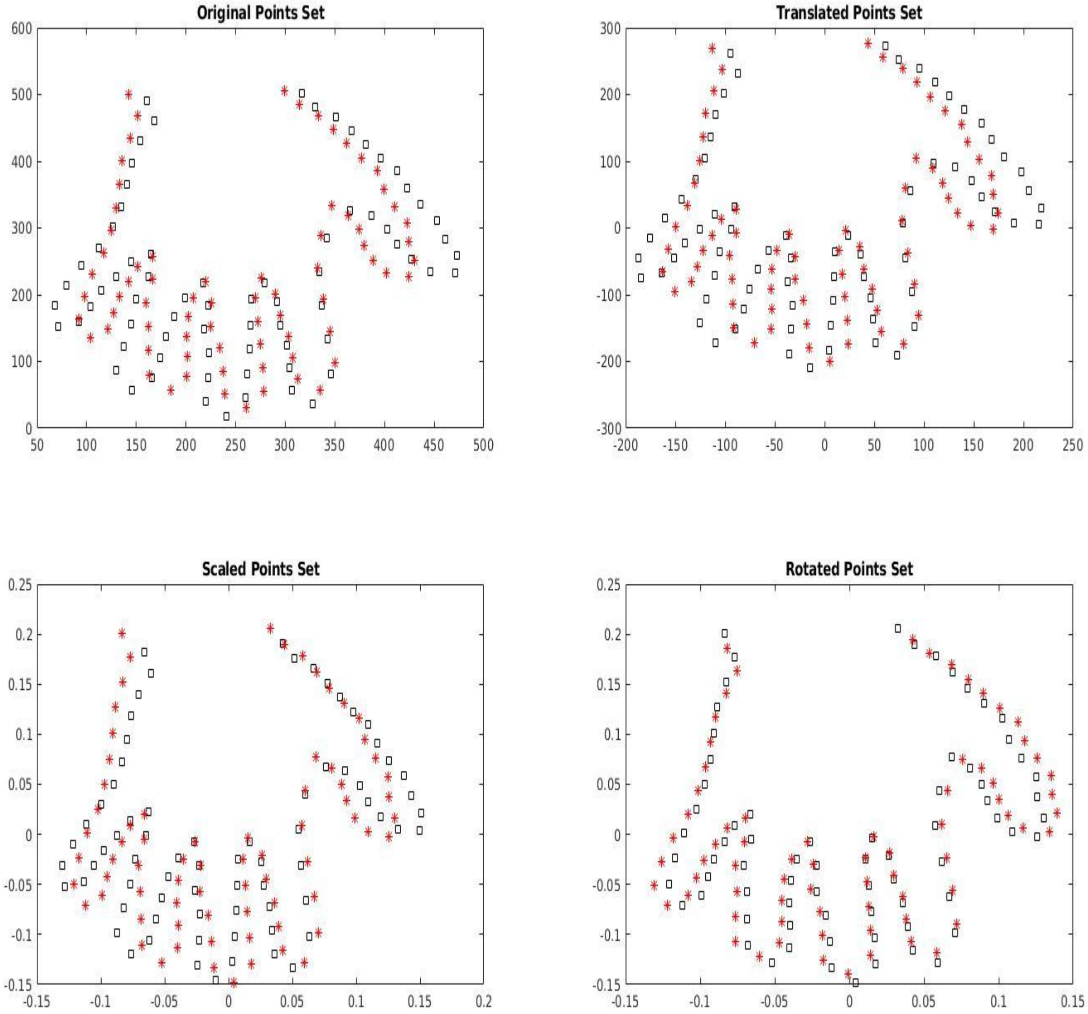
$$X \rightarrow SI \text{ with } S = sI \text{ and } s = \frac{\sum_{i=1}^{N} \mathbf{x}_i'^T \mathbf{x}_i}{\sum_{i=1}^{N} \mathbf{x}_i^T \mathbf{x}_i}$$

Equation for Rotation:

- form the product of the coordinate matrices XY^T
- calculate its SVD as $XY^T = USV^T$
- calculate the rotation matrix as $R = VU^T$



Observation for starpoints



Observation for handpoints

Contents

```
clear all; close all; clc;
```

Function Implemtation of Image :: Translation, Rotation and Interpolation

```
% I = imread('lena.jpeg');
% myTranslation(I, 45, 30);

% J = imread('cameraman.tif');
% myRotation(J, -30, 'bilinear');

% Function Implemtation of Image :: Projective Transformation

% Input = imread('plate_side.jpg');
% Ref = imread('plate_reference.jpg');
% projectiveTransform(Input, Ref);
```

Function Implemtation of Image :: Procrustes Analysis

```
M = load('star_points.mat');
myProcrustes(M.input_points,M.base_points);
```

Published with MATLAB® R2018a

Contents

```
function [O] = myTranslation(I, tx, ty)
```

```
%MYTRANSLATION Function shifts the image vertically by tx pixels and horizontlly by ty pixels.
%   Input::  I  - Input Image    tx - Vertical Shift     ty - Horizontal Shift
%
%   Output::  O - Translated Image
```

Function starts here

```
O = uint8( zeros(size(I,1), size(I,2), size(I,3)));
[r,c,~] = size(O);
for i=1:r
    for j=1:c

        % Shifted vertically by tx and horizontlly by ty
        O(i+tx:end, j+ty:end, :)  =  I(i:end-tx,j:end-ty,:);

    end
end

% Display the Resultant of Translated Image
figure,
subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(O), title('Translated Image');
```

```
Not enough input arguments.

Error in myTranslation (line 8)
O = uint8( zeros(size(I,1), size(I,2), size(I,3)));
```

```
end
```

Published with MATLAB® R2018a

Contents

```
function [O] = myRotation(I, ang, method)
```

```
%MYROTATION rotates the image from its centre with an angle 'theta'.
%  Input :: Input Image     ang  - Angle of Rotation    method  - Interpolation method {'nearest', 'bilinear'}

%  Output::  O - Rotated Image
```

Function starts here

```
I = im2double(I);

theta=(ang)*pi/360; % Angle of Rotation

[r, c] = size(I); % Size of the Image

O = zeros(size(r,c));

midx = r./2; % Midpoint of x-axis
midy = c./2; % Midpoint of y-axis

for i=1:r
    for j=1:c

        % Rotation steps (based on the Given Formula)
        x=(i-midx)*cos(theta)-(j-midy)*sin(theta);
        y=(i-midx)*sin(theta)+(j-midy)*cos(theta);
        x1=round(x)+midx;
        y1=round(y)+midy;

        % Interpolation steps
        if (1 <= x1 && x1 <= r && 1 <= y1 && y1 <=c)
            top = floor(x1);         bottom = top+1;
            left = floor(y1);       right = left+1;
            switch method % Select any one method to interpolate
                case 'nearest' % Nearest Neighbour Interpolation
                    if (bottom <= r && right <= c)
                     inten_1 =  I(top, left);    inten_2 =  I(bottom, left);
                     inten_3 =  I(top, right);   inten_4 =  I(bottom, right);
                     inten = (inten_1 + inten_2 + inten_3 + inten_4)/4;
                    end

                case 'bilinear' % Bilinear Interpolation
                  if (bottom <= r && right <= c)
                     inten_1 =  I(top, left);  inten_2 =  I(bottom, left);
                     leftInten = (x1-top) * (inten_2 - inten_1) + inten_1;

                     inten_3 =  I(top, right); inten_4 =  I(bottom, right);
                     rightInten = (x1-top) * (inten_4 - inten_3) + inten_3;

                     inten = (y1 - left) * (rightInten - leftInten) + leftInten;
                  end
            end
        else
            inten = 0; % Parts other than input image appears black
        end
        O(i,j)=inten; % Store the output image
    end
end

% Display the Result
figure,
subplot(1,2,1),imshow(I), title('Original Image of cameraman');
subplot(1,2,2),imshow(O), title('Rotated Image of cameraman');
```

```
Not enough input arguments.

Error in myRotation (line 9)
I = im2double(I);
```

```
end
```

Published with MATLAB® R2018a

Contents

```
function [Projected_image] = projectiveTransform(I, ref)
```

```
%MYPROCRUSTES Function that stretches or contracts the input points set to fit the reference points set.
%   Input    I - Input image     Y - Reference image
%
%   Output   O - Output Image (after projective transformation)
```

Function starts here

```
% Get coordinates of input and reference image
figure, imshow(I), title('Input Image of plate');
[x,y] = ginput(4);

figure, imshow(ref), title('Reference Image of the plate');
[x1, y1] = ginput(4);

X = [x,y];
Y = [x1,y1];

% Performing Projective Transformation [fitgeotrans is same as cp2tform]
T = cp2tform(X, Y, 'projective');

% Applying Transformation on the input image
Projected_image = imtransform(I, T);
figure, imshow(Projected_image), title('Projective Transformation');

% Cropping the Transformed image
[~, rect] = imcrop(Projected_image);
Img_crop = imcrop(Projected_image, rect);
figure, imshow(Img_crop), title('Croped Image (After Transformation)');
```

```
Not enough input arguments.

Error in projectiveTransform (line 10)
figure, imshow(I), title('Input Image of plate');
```

```
end
```

Published with MATLAB® R2018a

Contents

```
function [Z] = myProcrustes(X,Y)
```

```
%MYPROCRUSTES Function that stretches or contracts the input points set to fit the reference points set.
%   Input
%       X - Input points set
%       Y - Reference soints set
%
%   Output
%       Z - Fitted points set
```

Function starts here

```
figure,
subplot(2,2,1), plot(X(:,1),X(:,2),'ks',Y(:,1),Y(:,2),'r*'), title('Original Points Set');
```

```
Not enough input arguments.

Error in myProcrustes (line 13)
subplot(2,2,1), plot(X(:,1),X(:,2),'ks',Y(:,1),Y(:,2),'r*'), title('Original Points Set');
```

Translation (to fix the centre of both points set at the origin)

```
x = X - mean(X,1);
y = Y - mean(Y,1);

subplot(2,2,2), plot(x(:,1),x(:,2),'ks',y(:,1),y(:,2),'r*'), title('Translated Points Set');
```

Scaling (to stretch the points set to the same scale)

```
sx = sum(x.^2,1);
sy = sum(y.^2,1);
sx=sum(sx);
sy=sum(sy);

normX = sqrt(sx);
normY = sqrt(sy);

% Scale to equal (unit) norm
x = x / normX;
y = y / normY;

subplot(2,2,3), plot(x(:,1),x(:,2),'ks',y(:,1),y(:,2),'r*'), title('Scaled Points Set');
```

Rotate (to impose input points set on reference)

```
Scaled = x * transpose(y);
[U, ~, V] = svd(Scaled);
R = V * transpose(U);

Z = R * x;

subplot(2,2,4), plot(y(:,1),y(:,2),'ks',Z(:,1),Z(:,2),'r*'),title('Rotated Points Set');
```

```
end
```

Published with MATLAB® R2018a